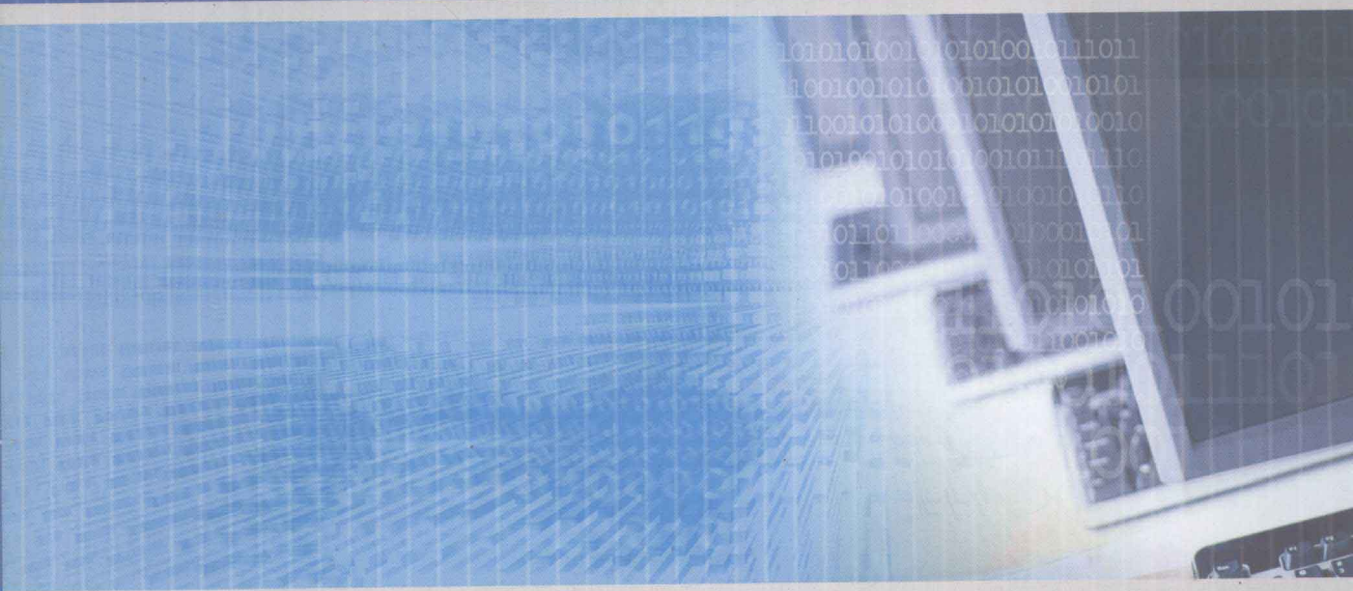
21 世纪全国高校应用人才培养信息技术类规划教材



# 汇编语言程序设计

刘丽莉 主 编

北京大学出版社
PEKING UNIVERSITY PRESS

21 世纪全国高校应用人才培养信息技术类规划教材

# 汇编语言程序设计

刘丽莉　主　编

北京大学出版社
PEKING UNIVERSITY PRESS

## 内 容 简 介

汇编语言是计算机专业的专业基础课，也是电子、通信等相关专业的计算机课程。本书介绍80X86汇编语言程序设计的方法和技术，共分为两个部分：第一部分为基础知识，介绍80X86 CPU的编程结构，汇编语言程序的格式和伪指令，80X86 CPU的寻址方式和指令系统；第二部分为编程方法，深入讨论分支程序、循环程序、子程序基本程序设计方法，以及以中断为主的I/O程序设计，其中包括宏指令、多模块连接技术、汇编语言与高级语言的混合编程、DOS和BIOS提供的常用中断调用，以及文件系统等内容。

本书结构清晰、内容循序渐进、例题多样、讲解详细，配有DEBUG下的截图进行说明，并有丰富的习题可供学生练习。本书包含上机实验内容，所有程序都经过运行验证。本书有配套的PPT课件可供下载。

本书适用于高等学校以及大、中专学校作为汇编语言程序设计课程的教材（含实验），也可作为其他专业相关课程的教材和参考书。

**图书在版编目(CIP)数据**

汇编语言程序设计/刘丽莉主编. —北京:北京大学出版社,2010.11
(21世纪全国高校应用人才培养信息技术类规划教材)
ISBN 978-7-301-17974-1

Ⅰ.①汇… Ⅱ.①刘… Ⅲ.①汇编语言—程序设计—高等学校—教材 Ⅳ.①TP313

中国版本图书馆CIP数据核字(2010)第205622号

电 子 信 箱: zyjy@pup.cn
印 刷 者: 三河市北燕印装有限公司
发 行 者: 北京大学出版社
经 销 者: 新华书店
787毫米×1092毫米 16开本 20印张 480千字
2010年11月第1版 2010年11月第1次印刷
定 价: 36.00元

# 前　言

汇编语言是面向机器的语言，用汇编语言编写的程序在时间和空间两个方面的效率都很高，汇编语言也是唯一能够利用计算机所有硬件特性并能对其直接控制的语言。在很多需要涉及硬件的开发场合，如实时控制程序、计算机操作系统内核、高级语言的编译程序、硬件驱动程序等，汇编语言更是有着高级语言无法替代的作用。

汇编语言因 CPU 的不同而不同。本书介绍 80X86 CPU 的汇编语言程序设计方法和技术。本书由基础知识（第 1 章至第 4 章）、编程方法（第 5 章至第 8 章）两部分构成。第 1 章对汇编语言做一简单介绍，并简要介绍计算机执行程序的过程，以及计算机的数制和码制。第 2 章为 80X86 CPU 的编程结构，主要介绍有关 CPU 的硬件知识，这是学习 80X86 汇编语言必须了解的。第 3 章以一个简单、完整的程序为例，介绍汇编语言程序的格式、常用伪指令和汇编语言的上机过程，以便在学习时能够尽早地开始上机编程。第 4 章为寻址方式和指令系统，寻址方式是掌握汇编语言的关键之一，汇编语言程序的编写在很大程度上取决于能否灵活而合理地运用寻址方式。第 5 章和第 6 章分别介绍分支程序、循环程序和子程序设计，其中包括多模块连接技术、宏指令和汇编语言与高级语言的混合编程。第 7 章介绍 I/O 和中断程序设计，包括 DOS 和 BIOS 提供的常用中断调用。第 8 章介绍文件系统。

本书的编写希望达到两个目的，一个是通过本书，使学生掌握汇编语言的编程方法、思路和技巧，并对计算机的底层编程有一定认识；另一个是通过对汇编语言的介绍，使学生理解计算机底层运行程序的机制，了解计算机的工作原理，为以后一些课程的学习（如操作系统、微机原理等）打下基础。本书所有内容的安排和讲解也是基于这样两方面的考虑。比如强调 CS 和 IP 寄存器的作用，比如在介绍子程序设计时，除了让学生能够使用 CALL 指令和 RET 指令编写子程序结构的程序，还要通过 CALL 指令和 RET 指令内部执行的操作，让学生明白计算机内部如何能够做到调用子程序，又如何能够从子程序返回主程序，子程序多层嵌套时为什么子程序返回不会乱套等问题。实际上，学完这门课程，学生也会对以前学过的 C 语言的一些概念有更深刻的理解（如指针），也会明白数组等数据结构在计算机内部是如何组织和表示的。

计算机专业是一个实践性很强的专业，汇编语言程序设计课程也是这样，在学习本课程时一定要多动手、多上机编程，这样才能提高编程能力和程序调试能力，才能达到事半功倍的效果。为了便于学习，一些内容的讲解和例题，都有在 DEBUG 下的截图。另外，本书包含上机实验内容，便于教师进行实践教学。

本书力求做到通俗易懂，概念清晰，内容循序渐进，例题多样，习题丰富。书中所有程序都经过上机运行验证。同时，本书有配套的 PPT 课件可供下载。

本书第 1 章、第 6 章由刘丽莉编写，第 2 章、第 3 章和附录由刘晋明编写，第 4 章、第 5 章和第 8 章由吕俊音编写，第 7 章由万春编写。全书由刘丽莉负责统编和定稿。

本书作为教材，广泛吸取了国内众多专家学者的研究成果，编写的主要参考书目附后，在此向各位作者表示谢意！编者虽然对书稿进行了多次修改，但由于水平有限，错误或不当之处敬请读者批评指正。

编　者

2010年10月

# 目　录

# 第1章 基础知识

**本章教学目标**

- 了解汇编语言的特点及应用场合；
- 了解计算机硬件的基本组成以及 CPU 对内存的访问过程；
- 理解计算机执行程序的过程及工作原理；
- 掌握数制与转换；
- 掌握计算机中数和字符的表示。

汇编语言是一个出现很早的语言，在本章中，我们首先介绍汇编语言的特点，以及我们为什么要学习汇编语言；其次，汇编语言不同于高级语言，它是直接在硬件之上工作的编程语言，因此我们要了解计算机的基本结构和工作原理；最后，再介绍汇编语言中经常涉及的数制和码制问题。

## 1.1 什么是汇编语言

### 1.1.1 机器语言与汇编语言

计算机最早使用的是机器语言。机器语言的指令都由二进制 0 或 1 编码组成。机器语言是计算机唯一可以直接识别并执行的语言，因此它的执行速度是最快的。由于机器指令的设计和执行跟硬件结构密切相关，因此每一类 CPU 都有自己的机器语言，它只认识自己的机器语言指令，机器语言没有通用性，用机器语言编写的程序也没有通用性，不能移植。

由于机器语言用一系列 0、1 表示所有的信息（指令和数据），因此用机器语言编写程序非常费力且枯燥、繁琐，而且程序不具有可读性，调试、修改都很困难。

机器语言使用一段时间后，编程人员很快就发现了机器语言的缺点，于是汇编语言出现了。汇编指令和机器指令的差别在于指令的表示方法上。汇编指令是机器指令便于记忆的书写格式，它采用容易记忆的符号和标记表示机器语言的指令，使程序具备了一定的可读性，也变得较为容易调试和修改。

例如，机器指令 1000100111011000 表示把寄存器 BX 的内容传送到 AX 中。用汇编语言则表示成 MOV AX，BX。这就容易识别多了。

因为汇编语言跟硬件密切相关，所以用汇编语言编写的程序也没有通用性。

但是，计算机只认识机器语言的指令，用汇编语言和其他高级语言编写的程序（称为源程序），必须用一个翻译程序（称为编译器）将它们翻译成机器语言程序（称为目标程序），计算机才能够执行。

但很快，编程人员又发现用汇编语言编程还是不够方便，于是又出现了高级语言。高级语言使用人们容易理解的符号和单词组成语句，语句间依照规定的语法规则构建程序。由于更接近于自然语言，因此高级语言程序的可读性很好，也容易调试和维护。不同于机器语言

和汇编语言，高级语言是独立于机器的程序设计语言，用它编写程序不需了解具体计算机硬件的内部逻辑，程序在不同的 CPU 上运行时基本不用修改，因此程序的可移植性好。

虽然高级语言程序容易编写，但是比起汇编语言的程序来说，其目标程序的效率不高，也就是说，高级语言的目标程序需要占用更大的存储空间、需要运行更长的时间。

### 1.1.2 汇编语言的特点及应用场合

汇编语言具有以下特点。

（1）面向机器的低级语言，通常是为特定的计算机或计算机系列专门设计的。

汇编语言的本质和机器语言接近，它只是用容易记忆的形式表示机器语言，因此跟机器语言一样，每一类 CPU 都有自己的汇编语言。

（2）保持了机器语言的优点，目标代码简短，占用内存少，执行速度快，是高效的程序设计语言。

这是汇编语言的一大优点。在对程序的空间和时间要求很高的场合，一般需要使用汇编语言。下面举例说明。

**例 1.1** 编程实现 c = a + b，并在屏幕上显示出结果。

C 语言程序如下：

```
#include "stdio.h"
void main( )
{int a,b,c;
       a =1;
       b =2;
       c =a +b;
       printf("c =% d \n",c);
       return 0;
}
```

编译后的目标文件达到 11.2 KB。

完成同样功能的 8086 CPU 的汇编语言程序如下：

```
DATA  SEGMENT
A           DB    ?
B           DB    ?
C           DB    ?
STRING      DB    'C = $'
DATA  ENDS
CODE  SEGMENT
MAIN  PROC FAR
      ASSUME CS:CODE, DS:DATA, ES:DATA
START:
       PUSH    DS
       SUB     AX,AX
       PUSH    AX
       MOV     AX,DATA
       MOV     DS,AX
       MOV     ES,AX
       MOV     A,1
       MOV     B,2
       MOV     AL,A
```

```
        ADD     AL,B
        MOV     C,AL
        LEA     DX,STRING
        MOV     AH,09
        INT     21H
        ADD     C,30H
        MOV     DL,C
        MOV     AH,2
        INT     21H
        MOV     DL,0AH
        INT     21H
        MOV     DL,0DH
        INT     21H
        RET
MAIN    ENDP
CODE    ENDS
        END     START
```

汇编后的目标文件只有 589 字节。

从上面的例子可以看出，高级语言编写的程序（源程序）看起来简短，但翻译成计算机能够执行的机器语言程序（目标程序）却很长，占用内存大，执行速度慢；汇编语言编写的程序（源程序）看起来长，但翻译成计算机能够执行的机器语言程序（目标程序）却很短，占用内存小，执行速度快。

（3）可以有效地访问、控制计算机的各种硬件设备，如磁盘、存储器、CPU、I/O 端口等。

这是汇编语言的另一大优点。汇编语言是能够利用计算机所有硬件特性并能直接控制硬件的唯一语言。因此在很多需要直接控制硬件的应用场合，一般都使用汇编语言，高级语言很难做到这一点。

（4）经常与高级语言配合使用，应用十分广泛。

汇编语言可以独立编写程序进行应用，也可以与高级语言配合使用。

根据以上特点，可以看出汇编语言的应用场合。汇编语言一般用在系统程序、效率代码和 I/O 驱动程序中。例如 70% 左右的系统软件使用或部分使用汇编语言编写；某些快速处理、位处理、访问硬件设备的程序和高级绘图程序等使用汇编语言编写。

### 1.1.3　汇编语言与高级语言的比较

汇编语言和高级语言各有优缺点。高级语言程序的可读性、可移植性好，编写和调试程序相对容易，编程效率高，但其编译产生的目标程序的效率却不高，也就是说，要占用更大的存储空间、要运行更长的时间。另外，高级语言很难直接对硬件加以控制。因此，在对程序的空间和时间要求很高的场合和在要求直接对硬件控制的场合，一般需要使用汇编语言编程（随着 C 编译器的性能越来越强大，并且 C 语言允许直接访问物理地址，能进行位操作，能实现汇编语言的大部分功能，因此，这些场合有时也用 C 语言编程）。这就是为什么汇编语言虽然是一门古老的语言，但直到现在仍然具有生命力，仍在学习、使用的原因。

# 1.2 硬件基础

## 1.2.1 计算机硬件的基本组成

计算机由软件和硬件组成。硬件就是构成计算机各部件的实体，是计算机中看的见、摸的到的部分。硬件是软件运行的基础。

自从1946年出现了世界上第一台计算机以来，计算机科学得到了迅猛的发展，计算机的硬件结构和软件系统都发生了巨大的变化。但是，就其硬件基本组成来说，至今仍未摆脱冯·诺依曼型计算机的设计思想，即计算机的硬件结构是由运算器、控制器、存储器、输入设备和输出设备五大基本部分组成，如图1-1所示。

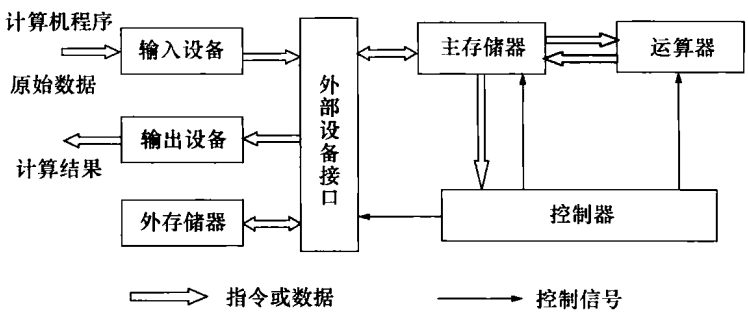



图1-1 计算机的硬件结构

冯·诺依曼型计算机的设计思想还包括计算机中的程序和数据采用二进制表示、存储程序与程序控制两方面。存储程序与程序控制的思想也正是计算机的工作原理，即人们把事先编好的程序和所需要的数据通过输入设备送到内存或外存中保存，这一步称之为存储程序；开始工作时，控制器从内存中逐一读取程序中的指令，并按照每条指令的要求执行所规定的操作，这称之为程序控制。

由于大规模和超大规模集成电路技术的发展，人们已将控制器和运算器集成在一个芯片上，称为中央处理器，在微机中称为微处理器（CPU）。图1-2是一台典型微型计算机的组成框图，它由微处理器、存储器、I/O（输入/输出）设备及其接口组成，各部分通过总线连接在一起，通过总线传送数据、地址和控制信息。

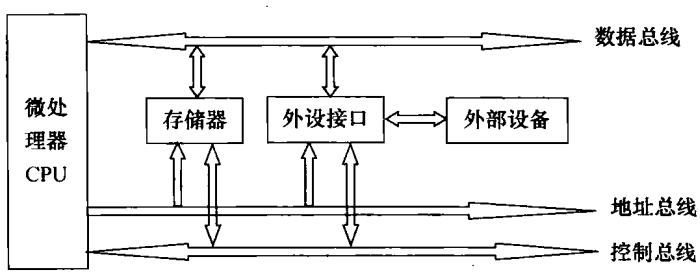



图1-2 微型计算机的组成框图

### 1.2.2　地址、数据和控制总线

所谓总线实际上就是连接计算机各个功能部件之间或计算机之间的一束公共信息传输线，它是计算机系统中传送信息的公共途径。总线（BUS）由地址总线（AB）、数据总线（DB）和控制总线（CB）组成。

1. 地址总线

地址总线不仅传送内存地址，而且还传送外设的地址。地址总线是单向的，地址都是由 CPU 送出。

CPU 通过地址总线来指定要访问的内存单元或外设的地址。可见，地址总线上能传送多少个不同的信息，CPU 就可以对多少个内存单元或外设进行寻址。

假设一个 CPU 有 1 根地址总线，则和其连接的内存最多只可能有两个单元，一个单元的地址为 0，另一个单元的地址为 1。假设一个 CPU 有 10 根地址总线，则和其连接的内存最多可以有 1024 个单元，其单元地址从 0000000000 到 1111111111。一个 CPU 有 N 根地址总线，则可以说这个 CPU 的地址总线的宽度为 N，这样的 CPU 最多可以寻找 $2^N$ 个内存单元。

2. 数据总线

数据总线进行数据的传送。数据总线是双向的，可以实现双方的数据传送。

CPU 与内存或外设之间的数据传送是通过数据总线来进行的。数据总线的宽度影响 CPU 和外界的数据传送速度。8 位数据总线一次可以传送一个 8 位二进制数据（即一个字节）。

3. 控制总线

控制总线传送控制信号。例如，CPU 通过控制总线向内存发出读、写信号；CPU 通过控制总线向外设发出启停、读、写等命令，来控制外设的操作；外设通过控制总线向 CPU 发出中断请求信号等。显然，控制总线中有些信号是 CPU 发出的，有些信号是发往 CPU 的，控制总线包括这两个方向的信号线。

有多少根控制总线，就意味着 CPU 提供了对外部器件的多少种控制。所以，控制总线的宽度决定了 CPU 对外部器件的控制能力。

### 1.2.3　CPU 对内存的访问

CPU 是计算机的核心部件。它控制整个计算机的运行并进行运算，要想让一个 CPU 工作，就必须向它提供指令和数据。指令和数据在内存中存放。外存（如硬盘、光盘等）不同于内存，外存上的数据或程序如果不读到内存中，就无法被 CPU 使用。

内存被划分为一个个的存储单元，每个存储单元从 0 开始顺序编号，这就是存储单元的地址。如果一个内存有 1024 个单元，则该内存要有 10 根地址线，来给出这 1024 个单元地址。

内存与 CPU 的连接见图 1-3。

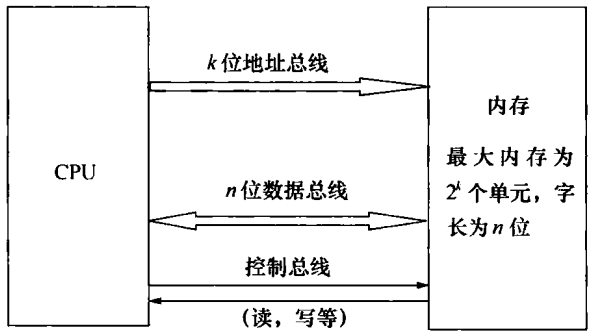



图 1-3 内存与 CPU 的连接

1. 对内存读操作

CPU 要对内存读，必须先在地址线上送出内存单元的地址，然后再发出“读”控制信号，这样经过一段时间（内存对地址进行译码选中相应单元，以及从该单元读出数据所需的时间。该时间反映了内存的读速度快慢），就会在数据线上出现所读单元中的数据，CPU 就可以取走进行处理了。

2. 对内存写操作

CPU 要对内存写，同样必须先在地址线上送出内存单元的地址，同时还要在数据线上送出要写入的数据，然后发出“写”控制信号，这样经过一段时间（内存对地址进行译码选中相应单元，以及把数据线上的数据写入从该单元所需的时间。该时间反映了内存的写速度快慢），就会把数据写入该内存单元中。

**说明：**在内存或外存中，指令和数据没有任何区别，都是二进制形式。所有信息在计算机中都是以 0、1 的形式存在。计算机中信息可分为指令和数据两类，指令告诉计算机做什么，数据是计算机处理加工的对象，数据可以是学生的成绩、内存单元的地址，也可以是声音、图像等形式的数据。因此，CPU 从内存读出的可能是指令，也可能是数据。后面大家会清楚 CPU 是如何区分它们的（即在 8086 CPU 中，寄存器 CS：IP 指向的是指令，其他都是数据）。

例如，二进制信息 1000100111011000，如果把它当数据，就是 89D8H（十六进制数据），如果把它当指令，就是指令 MOV AX，BX。

## 1.2.4 内存地址空间

在一台 PC 主机中，除内存外，还有其他一些存储单元。CPU 对外部设备不能直接控制，而是要通过相应的接口卡对这些设备进行控制，这些接口卡插在主机板的扩展插槽上。此外，这些接口卡上还有一些存储单元。

因此，从读写属性上看，内存分为两类：随机存储器（RAM）和只读存储器（ROM）。

而从功能和连接上看，内存又分成以下 3 种。

1. 随机存储器 RAM

随机存储器 RAM 是内存的主要部分，由主板上的 RAM 和插在扩展槽中的 RAM 两部分组成。

2. 装有 BIOS（Basic Input/Output System，基本 I/O 系统）的 ROM

BIOS 是由主板和各类接口卡（如显卡、网卡等）厂商提供的软件系统，可以通过它利用该硬件设备进行最基本的 I/O。在主板和某些接口卡上插有存储相应 BIOS 的 ROM。

例如，主板上的 ROM 中存储着系统 BIOS，显卡上的 ROM 存储着显卡的 BIOS。

3. 接口卡上的 RAM

某些接口卡上还有 RAM，用于暂存数据。典型的是显卡上的 RAM，一般称为显存。显卡随时将显存里的数据显示在显示器上。也就是说，如果我们把想显示的数据写入显存，那么数据就会显示在显示器上。

上面这些物理上独立的存储器，在 CPU 访问它们的时候，是把它们当成一个逻辑上统一的存储器看待的。每个物理上的存储器在这个逻辑存储器中都占有一段单元。例如，彩显的显存空间从 B8000H 开始，单显的显存空间从 B0000H 开始。这样，当我们往 B8000H 开始的显存空间写入数据时，该数据就被显示在显示器上。

内存地址空间的大小受 CPU 地址总线宽度的限制。8086 CPU 的地址总线宽度为 20 根，因此可以定位 $2^{20}$ 个（1 M）内存单元（每个单元存储 1 个字节），也就是说 8086 CPU 的地址空间的大小为 1 MB。

不同计算机系统的内存地址空间的分配情况是不同的，图 1-4 是 8086 PC 的内存地址空间分配的基本情况。

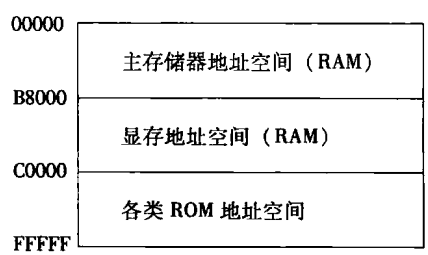



图 1-4　8086 PC 的内存地址空间分配

## 1.3　计算机执行程序的过程

下面以表 1-1 中的程序为例，说明计算机程序的执行过程，由此可了解到计算机的工作原理。

表 1-1 计算 X + Y 的程序

| 指令序号 | 汇编语言指令 | 指令存储地址 | 机器指令 | 指令功能 |
|---|---|---|---|---|
| 1 | MOV AL, X | 2000 | A00100 | 将 X 单元（0001 号）中的内容传送到 AL 寄存器中 |
| 2 | ADD AL, Y | 2003 | 02060200 | 将 Y 单元（0002 号）中的内容加到 AL 寄存器中 |
| 3 | MOV SUM, AL | 2007 | A20300 | 把 AL 寄存器中的和传送到 SUM 单元（0003 号）中 |
| 4 | HLT | 200A | F4 | 停机 |

首先要输入程序。通过键盘和鼠标这些输入设备，我们把程序送入计算机中。保存程序时，程序将以文件的形式存放在外存上。然后我们对该源程序进行编译，产生目标文件（.EXE 文件），该目标文件也在外存上。当我们在 Windows 下双击这个 .EXE 程序的图标，或 DOS 下在命令行上输入要运行的这个 .EXE 程序的文件名时，操作系统将负责把这个目标程序从外存调到内存（操作系统也是以文件的形式存储在外存上，外存上的程序只有调入到内存后才能执行，那么操作系统又是被谁，又是如何调入到内存并执行的呢？这个问题将在后面的章节中解释），安排在内存的一个合适区域中（如表 1-1 所示，假设程序存放在 2000～200A 这段空间），并把这个区域的首址（2000）送到程序计数器 PC 中。程序计数器 PC 是计算机中很重要的一个寄存器，它始终给出要执行的下一条指令的地址，CPU 就是按照它的指向去取指令执行的。每个计算机的 CPU 中都有这样功能的一个寄存器，但名字可能起的不一样（在 8086 CPU 中，与 PC 寄存器作用相当的是 CS 和 IP 这两个寄存器）。因此，操作系统把这个程序的首址（2000）送到 PC 中，也就是让 CPU 从内存的 2000 单元开始执行程序。

接下来开始程序的执行。

1. 取第一条指令并执行

（1）取指令并分析指令。

第一条指令是一个三字节指令。首先计算机把 PC 的值 2000 送到地址总线上，并发出读信号，这样就从内存的 2000 号单元取出第一条指令的第一个字节（PC 将自动加 1，变成 2001），该字节实际上是本条指令的操作码字段，通过数据总线将它从内存送到 CPU 中，经 CPU 中的指令译码器分析译码后，得知该指令是“传送”指令，要把内存的“某个单元”的内容送到 AL 寄存器，“某个单元”的地址即由指令的第二个和第三个字节构成（存放时高字节放在后面，低字节放在前面，即单元地址为 0001），第二个和第三个字节就是本指令的地址码字段；接着继续重复刚才的过程，按照 PC 的指向到内存 2001 单元取出第二个字节（即 01），PC 的值变成 2002；再从 2002 单元取出本指令的第三个字节（即 00），PC 的值变成 2003，这时，就得到了地址 0001。到此为止，第一条指令取出并分析完毕。

（2）执行指令。

CPU 中的执行部件接收到来自指令译码器的译码信号“取数并传递”后，则转入